CS3723 Python Pgm 5 (20 pts) Python - due date is 11/26

Business Problem

You have been tasked with executing source code for a simple programming language named BEEP. The business plans to use BEEP to provide simple transformations.

Summary of Program 5 versus Program 6

In program 5, you will read a file that contains many BEEP source code text lines. This program will read the text lines and place them in a homogeneous array (i.e., list). It will populate associative arrays (dictionaries) with information based on the BEEP source code.

In program 6, you will execute the BEEP source code.

Associative arrays

varTypeD	Contains the **data type** for each **variable**
varValueD	Contains the **value** for each **variable**
labelD	uses a **label** as the key and stores the **subscript** of the statement

BEEP statements

VAR *dataType variableName initialValue*

This declares the specified *variableName* to be of the specified data type. Use the *variableName* as the key in both the varTypeD and varValueD dictionaries. If specified, the *initialValue* is the initial value for the variable *variableName*. Examples:

```
VAR int count 0
VAR string name
VAR string greeting "Hello"
```

# *comment*

label*: *statement

The subscript for this line of code is stored in the labelD dictionary.

***statement* is one of:**

ASSIGN *variableName expression*
IF *expression label*
PRINT *varLiteral1 varLiteral2 …*
GOTO *label*

expression is one of:

varLiteral	
* *varLiteral varNumber*	return a string with *varLiteral* replicated *varNumber* times
+ *varNumber1 varNumber2*	return the sum of the values
- *varNumber1 varNumber2*	return the difference of the values (*varNumber1* minus *varNumber2*)
> *varNumber1 varNumber2*	return True if *varNumber1 > varNumber2;* this is a numeric comparison
>= *varNumber1 varNumber2*	return True if *varNumber1 >= varNumber2;* this is a numeric comparison
& *varLiteral1 varLiteral2*	return the concatenation of the two strings

Program Requirements

1. You will be provided an input file. Your program should be passed the name of that file as a command argument:
   ```
   python3 p5Driver.py p5Input.txt
   ```
2. Your Python source code must be separated into **multiple functions** and **source files**:

p5Driver.py This is the main driver for your code. The program is passed the name of the BEEP source code file. It reads and prints all the BEEP source code lines (showing line numbers) and places them in a list.

As you are reading the lines:

- If the line begins with a token that ends with a colon, it is a label. Save labels and their line number (relative to 1) in the labelD dictionary. Labels should be saved as uppercase values.
- if the first non-label token is a VAR command, invoke **declareVar**(tokenM, varTypeD, varValueD). Variable names should be saved as uppercase values.

It also adds labels to the labelD dictionary.

As shown below, after reading the BEEP source code, it should print the variables and their data types (and values if provided). Additionally, it should print the labels and their locations.

In program 6, there will be a separate loop to execute the BEEP code.

p5Dict.py This code provides the **declareVar, printVariables,** and **printLabels** functions**.** It can have other functions if needed. You should print the variables and labels in ascending order by the names.

To import code from another file, use a Python statement like this:

```
from fileName import funcName1, funcName2
```

Note the actual file would be named *fileName*.py. Inotherwords, the filename specified on the **from** would be without the **.py.**

3. If the same variable name is encountered in multiple VAR statements, simply re-declare the variable with its new type and possibly value. Variable names in BEEP are **not** case sensitive.
4. If the same label is encountered at the beginning of two or more statements, show an error message (see below) and the label. Labels are **not** case sensitive. Print a message showing its first line number and the other one:
```
***Error: label 'LOOP' appears on multiple lines: 9 and 20
```
Do **not** terminate.
5. Documentation:
   - p5Driver.py must have documentation describing purpose, input, and output.
   - p5Dict.py must have documentation for each function describing its purpose, parameters, and returns. Please document the purpose of each conditional statement.
6. **Any use of code from another web site will result in a 0 on this assignment, may result in an F for this course, and may cause you to be expelled from UTSA.**
7. Execute your code on a fox server using **python3**
8. Turn in a zip file named *LastnameFirstname*.zip which contains:
   p5Driver.py
   p5Dict.py
   p5Out.txt – contains the output generated by your program for **p5InputC.txt**.
   Your zip file must not contain any directories.
9. Your program must work with each of the input files, some of which should cause you to print error messages. You only have to turn in the output for the input file specified above. The TA will also run you code on the other input files.

Sample Input:

```
VAR int count 0
VAR string result
VAR string symbol "ho"
VAR int tick 3
VAR int limit 10
VAR string greeting "hello..there"
VAR int iter 4
PRINT "begins..."
PRINT "Top:...count=" count "tick=" tick "symbol=" symbol
ASSIGN count + count 1
IF > tick limit pastlimit
    ASSIGN result * symbol tick
    PRINT result
    ASSIGN tick + tick 1
    GOTO afterIf
pastlimit: PRINT "***tick...reached...limit:" tick
afterIf: PRINT "Bot:...count=" count "tick=" tick "symbol=" symbol
PRINT "EndPgm"
```

Sample Output:

```
BEEP source code in p5InputA.txt:
  1. VAR int count 0
  2. VAR string result
  3. VAR string symbol "ho"
  4. VAR int tick 3
  5. VAR int limit 10
  6. VAR string greeting "hello..there"
  7. VAR int iter 4
  8. PRINT "begins..."
  9. PRINT "Top:...count=" count "tick=" tick "symbol=" symbol
 10. ASSIGN count + count 1
 11. IF > tick limit pastlimit
 12.     ASSIGN result * symbol tick
 13.     PRINT result
 14.     ASSIGN tick + tick 1
 15.     GOTO afterIf
 16. pastlimit: PRINT "***tick...reached...limit:" tick
 17. afterIf: PRINT "Bot:...count=" count "tick=" tick "symbol=" symbol
 18. PRINT "EndPgm"
Variables:
    Variable     Type     Value
    COUNT        INT      0
    GREETING     STRING   hello..there
    ITER         INT      4
    LIMIT        INT      10
    RESULT       STRING
    SYMBOL       STRING   ho
    TICK         INT      3
Labels:
    Label        Statement
    AFTERIF      17
    PASTLIMIT    16
```